Chapter 2

Problem Statement

In this chapter we will state the governing equations under consideration, including the Navier-Stokes equations for incompressible fluid flow. The equations governing the mass and momentum balance are given in Section 2.1. An overview of constitutive relations relevant to the current study is presented in Section 2.2.

2.1 Equations of Motion for Incompressible Fluid Flow

We consider a viscous, incompressible fluid occupying at an instant $t \in (0, T)$ a bounded region $\Omega_t \subset \mathbb{R}^{n_{\text{sd}}}$, with boundary Γ_t, where n_{sd} is the number of space dimensions. The velocity and pressure, $\mathbf{u}(\mathbf{x}, t)$ and $p(\mathbf{x}, t)$, are governed by the momentum and mass balance equations:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \mathbf{f}\right) - \nabla\cdot\boldsymbol{\sigma} = \mathbf{0} \qquad \text{on } \Omega_t \quad \forall t \in (0, T), \tag{2.1}$$

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{on } \Omega_t \quad \forall t \in (0, T), \tag{2.2}$$

where ρ is the fluid density, assumed to be constant, and $\mathbf{f}(\mathbf{x}, t)$ is an external, e.g., gravitational, force field. The closure is obtained with a constitutive equation relating the stress tensor $\boldsymbol{\sigma}$ to velocity and pressure fields. These constitutive equations are presented in the next section. When the Newtonian constitutive model is used, the equations (2.1–2.2) are referred to as Navier-Stokes equations for an incompressible fluid. Both the Dirichlet and Neumann-type boundary conditions are taken into account, represented as:

$$\mathbf{u} \cdot \mathbf{e}_d = g_d \qquad \text{on } (\Gamma_t)_{\boldsymbol{g},d}, \quad d = 1 \ldots n_{\mathrm{sd}}, \tag{2.3}$$

$$\mathbf{n} \cdot \boldsymbol{\sigma} \cdot \mathbf{e}_d = h_d \qquad \text{on } (\Gamma_t)_{\boldsymbol{h},d}, \quad d = 1 \ldots n_{\mathrm{sd}}, \tag{2.4}$$

where $(\Gamma_t)_{\boldsymbol{g},d}$ and $(\Gamma_t)_{\boldsymbol{h},d}$ are complementary subsets of the boundary Γ_t, and $\{\mathbf{e}_d\}_{d=1}^{n_{\mathrm{sd}}}$ is a basis in $\mathbb{R}^{n_{\mathrm{sd}}}$. Note that the decomposition of Γ_t may be different for each of the basis vectors $\mathbf{e}_d$. In practice, the bases often coincide with the local directions normal and tangential to the boundary. In three dimensions, at each point of the boundary Γ_t we define one unit vector normal to the surface, as well as two unit vectors defining the plane tangent to the boundary. These vectors are denoted by $\mathbf{n}$, $\mathbf{t}$ and $\mathbf{b} = \mathbf{t} \times \mathbf{n}$, respectively. We now list some frequently encountered choices of boundary condition sets:

- Inflow or no-slip boundary used to specify all components of the velocity, applied at the inflow boundaries and solid obstacles and walls. Using the previously introduced definitions, we can state that:

$$(\Gamma_t)_{\mathit{inflow}} \subset (\Gamma_t)_{\boldsymbol{g},\mathbf{n}} \cap (\Gamma_t)_{\boldsymbol{g},\mathbf{t}} \cap (\Gamma_t)_{\boldsymbol{g},\mathbf{b}}. \tag{2.5}$$

- Symmetry boundary used to specify zero normal component of velocity and zero shear stresses, applied at the boundaries which represent flow symmetry lines. It can be symbolically written as:

$$(\Gamma_t)_{\mathit{symmetry}} \subset (\Gamma_t)_{\boldsymbol{g},\mathbf{n}} \cap (\Gamma_t)_{\boldsymbol{h},\mathbf{t}} \cap (\Gamma_t)_{\boldsymbol{h},\mathbf{b}}. \tag{2.6}$$

- Outflow boundary used to specify only the (zero or otherwise) stress components, normally applied at outflow boundaries or free surfaces. It is also referred to as the traction-specified boundary and can be defined as:

$$(\Gamma_t)_{\mathit{outflow}} \subset (\Gamma_t)_{\boldsymbol{h},\mathbf{n}} \cap (\Gamma_t)_{\boldsymbol{h},\mathbf{t}} \cap (\Gamma_t)_{\boldsymbol{h},\mathbf{b}}. \tag{2.7}$$

In the two-dimensional case only the $\mathbf{n}$ and $\mathbf{t}$ vectors are defined at the boundary. The definitions (2.5), (2.6) and (2.7) still apply, with the boundary subsets $(\Gamma_t)_{\bullet,\boldsymbol{b}}$ dropped.

The initial condition consists of a divergence-free velocity field specified over the entire domain:

$$\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0, \quad \nabla \cdot \mathbf{u}_0 = 0 \quad \text{on } \Omega_0. \tag{2.8}$$

From a numerical point of view the coupled equations (2.1) and (2.2) have a number of interesting and challenging features. The momentum equation is non-linear due to the presence of the advection term. Thus an iterative method, such as the Newton-Rhapson algorithm, have to be used in the solution process. The momentum equation has a variable character depending on the properties of the modeled fluid. Its behavior ranges from diffusive (for highly viscous fluids and creeping flows), to advective (for inviscid fluids or high-speed flows). Another important feature is the presence of the incompressibility constraint (2.2) and related pressure variable playing the role of a Lagrange multiplier enabling the flow to satisfy that constraint. Many methods of numerically solving the incompressible Navier-Stokes equations exist – see Gresho [1] for a review. Here we will apply the mixed formulations with the velocity and the pressure acting as primary variables.

2.2 Constitutive Relations

The present work deals with both Newtonian and non-Newtonian fluids. The former include water and most gases. The latter include fluids with a complicated molecular structure, e.g., polymers, emulsions or rubber, and have a great deal of industrial significance.

In the constitutive equation for a Newtonian fluid, the deviatoric stress (also referred to as extra-stress or non-isotropic stress component) is assumed to be proportional to the fluid rate of strain tensor which is defined as:

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right). \tag{2.9}$$

Under this assumption the stress tensor for a fluid with kinematic viscosity μ is defined as follows:

$$\begin{aligned} \boldsymbol{\sigma} &= -p\mathbf{I} + \mathbf{T}, \\ \mathbf{T} &= 2\mu\boldsymbol{\varepsilon}(\mathbf{u}). \end{aligned} \tag{2.10}$$

Modifications of the basic Newtonian relation (2.10) include shear-thickening, as well as shear-thinning culminating in plasticity.

Viscoelastic fluids exhibit dependence of the stress not only on the instantaneous rate of strain, but also on the strain history. For the upper-convected Maxwell fluid, also called the Maxwell-B fluid, the constitutive equation acquires an evolutionary character:

$$\begin{aligned} \boldsymbol{\sigma} &= -p\mathbf{I} + \mathbf{T}, \\ \mathbf{T} + \lambda\overset{\triangledown}{\mathbf{T}} &= 2\mu\boldsymbol{\varepsilon}(\mathbf{u}), \end{aligned} \tag{2.11}$$

where the $\overset{\triangledown}{\mathbf{T}}$ denotes an upper-convected derivative:

$$\overset{\triangledown}{\mathbf{T}} = \frac{\partial \mathbf{T}}{\partial t} + \mathbf{u}\cdot \nabla\mathbf{T} - \left(\nabla\mathbf{u}\,\mathbf{T} + \mathbf{T}\,(\nabla\mathbf{u})^T\right). \tag{2.12}$$

The upper-convected Oldroyd model, also known as the Oldroyd-B model, includes the Newtonian and Maxwell models, and covers the cases in which an elastic fluid obeying the Maxwell relation is mixed with a fluid governed by a Newtonian law. This corresponds to a situation in which an elastic polymer with viscosity μ_1 is dissolved in a viscous solvent with viscosity μ_2:

$$\begin{aligned} \boldsymbol{\sigma} &= -p\mathbf{I} + \mathbf{T}, \\ \mathbf{T} &= \mathbf{T}_1 + \mathbf{T}_2, \\ \mathbf{T}_1 + \lambda\overset{\triangledown}{\mathbf{T}}_1 &= 2\mu_1\boldsymbol{\varepsilon}(\mathbf{u}), \\ \mathbf{T}_2 &= 2\mu_2\boldsymbol{\varepsilon}(\mathbf{u}), \\ \mu_1 + \mu_2 &= \mu. \end{aligned} \tag{2.13}$$

The Maxwell fluid is extremely difficult to handle numerically, and this point is expanded upon in Section 4.3. The main feature of (2.11) is the convective character of the stress evolution equation. The difficulty is lessened considerably by even a small addition of the Newtonian solvent in an Oldroyd-B fluid. Even so, one may expect the problems normally associated with an advective systems to arise when discretization of the constitutive equation is performed. Additional feature introduced by the viscoelastic equations is their non-linearity. From the numerical point of view, that property is a strong argument for solving the equations for stress and velocity in a coupled manner. The Newtonian model for the stress is easily incorporated into the momentum equation (2.1) itself to yield some of the most commonly seen forms of the Navier-Stokes equations. The viscoelastic models on the other hand, necessitate treatment of the extra stress as a separate variable.

Chapter 3

Space-Time Velocity-Pressure Formulation

In this chapter a space-time velocity-pressure formulation is presented. Its inherent ability to handle arbitrary deformations of the physical domain makes it an attractive alternative to existing methods. A general background on the established numerical methods dealing with free surface problems is given in Section 3.1. The present method is introduced in Section 3.2, and further discussed in Sections 3.3 and 3.4. The chapter is concluded with Sections 3.5 and 3.6 covering the implementational issues. The discussion in this chapter is restricted to fluids governed by a Newtonian constitutive law.

3.1 Background

When faced with the fluid dynamics problems involving free surfaces, moving interfaces and deforming domains in general, a decision tree regarding the choice of a numerical method shown in Figure 3.1 may be the one to follow.

The Marker-And-Cell (MAC) method developed by Harlow and Welch, and described in an exhaustive report [2], has stood its own against the more established finite element and finite difference approaches in this particular area. The numerous

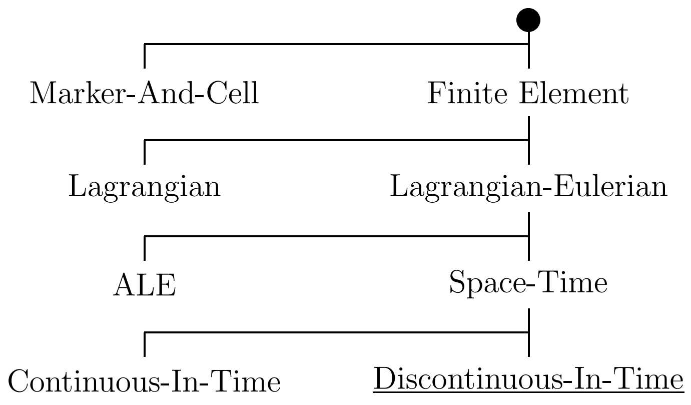

Figure 3.1. Numerical methods applicable to problems involving deforming domains: author's decision tree (see text for explanation of terms).

examples presented in [2] attest to the method's extraordinary flexibility. The concept of a mesh covering the maximum possible domain, but with computations being performed only for the cells containing marker particles, leads to efficient implementations. Since the tracer particles are indistinguishable from one another, the joining and separation of fluid parts – a source of mesh generation nightmares in the finite difference/element approaches – can be effortlessly simulated. Yet the MAC method is not free of shortcomings. Various stress effects at the free surface, such as surface tension and contact angles, have been notoriously difficult to incorporate, requiring complex surface fitting techniques such as those introduced in [3]. The method can suffer from stability problems which may be difficult to detect. That is, apparently reasonable particle distribution may be an artifact of gross approximation errors. Some modification were proposed to stabilize the MAC method in [4]. Another variation of the MAC concept replaces the discrete marker particles with a continuous fractional volume of fluid (VOF) function, as described in [5].

The extensive pool of knowledge accumulated about the finite difference and finite element methods in the context of fixed domain problems makes these more standard approaches also attractive. Here, we will concentrate on the finite element approach.

It should be noted that some of the finite element methods reviewed in this section possess a finite difference counterpart. For a finite element method to be applicable to problems involving deforming domains, it needs to have one important feature which may be omitted in the case of problems involving fixed domains only. The necessity of accurately following the deforming boundary of the domain requires some degree of Lagrangian description to be present in the formulation, at least in the vicinity of the boundary. Since the use of a Lagrangian viewpoint is unavoidable, some researchers consider fully Lagrangian formulation for the entire domain [6, 7]. The simplicity of such solution is offset by several difficulties. The flow field in the interior of the domain may exhibit a large amount of circulation, such as local vortices, which is often unrelated to the motion of the boundary. The fully Lagrangian approach may result in unnecessary mesh distortion and tangling in such interior regions, even for moderate or null displacements of the boundary. On the other hand, internal circulation and shear are handled without difficulty by an Eulerian description.

These facts provided a strong motivation for development of methods capable of combining the Lagrangian and Eulerian approaches in the same domain. The Arbitrary Lagrangian-Eulerian (ALE) formulation was initially stated in the finite difference context by Hirt [8], and eventually adopted also in the finite element community – see [9,10] and references contained therein for a detailed description. In the modern ALE approach, velocities of the nodes of the computational mesh explicitly enter the momentum equation, which is written over a reference domain. This domain may or may not coincide with either material or spatial domains. Nodal velocities may be either prescribed a priori, or be a function of the fluid velocity (most notably at the free surface). The need for including three different coordinate systems in the formulation makes it quite complex and lowers its appeal. Nevertheless, the ALE method has been used to solve a number of interesting problems, including those quoted in [10] and [11].

Another avenue was pioneered by Jamet and Bonnerot [12, 13] as a means of solution of Euler equations and Stefan-type problems. The method uses finite element discretization in both space and time to automatically account for the deformation of the computational domain. This approach was later applied to shallow water equations by Lynch and Gray [14] and flows of incompressible fluids by Frederiksen and Watts [15]. Improving on the continuous-in-time interpolations used in the applications mentioned above, Jamet [16] proposed the use of discontinuous-in-time functions for a model parabolic problem. Here for the first time the numerical method was presented together with analytic proofs of stability and estimates of accuracy.

In a parallel development, the discontinuous-in-time space-time methods have been gaining popularity for fixed domain problems, as an alternative to the more common semidiscrete (finite elements in space, finite differences in time) approach. The advantages, such as potential for selective temporal refinement and possibility of obtaining rigorous convergence predictions, were discussed by Hughes and Hulbert in [17] in the context of elastodynamics. Considerable attention was given to the space-time method for fixed domains by Hansbo et al. [18]. Their characteristic streamline diffusion method takes advantage of the possibility of unstructured mesh across the time step to reduce advective effects by moving interior nodes approximately along the characteristic directions, followed by remeshing.

The potential of the discontinuous-in-time space-time formulation as a means of simulating flows of fluids in a deforming domain has been exploited in [19, 20]. Later applications of the technique are described in [21, 22]. It has also been used to conduct a large-scale studies of incompressible flows past oscillating airfoils and cylinders [23, 24]. More recently, the same method, with a particular mesh moving scheme conforming to the characteristic streamline diffusion ideology, has been applied to deforming domain problems by Hansbo [25]. Finally let us note that similar approach can be equally successful for compressible flow simulation [26]. The remainder of this chapter recapitulates the method introduced in [19, 20], and discusses mesh motion and related aspects. The discontinuous-in-time space-time technique considered here may be expected to retain the detailed accuracy analyses carried out for its fixed domain counterpart, while providing an extremely elegant way of accounting for the displacements of the reference domain, i.e., the finite element mesh.

Note that in addition to the direct modeling of the governing equations, the free

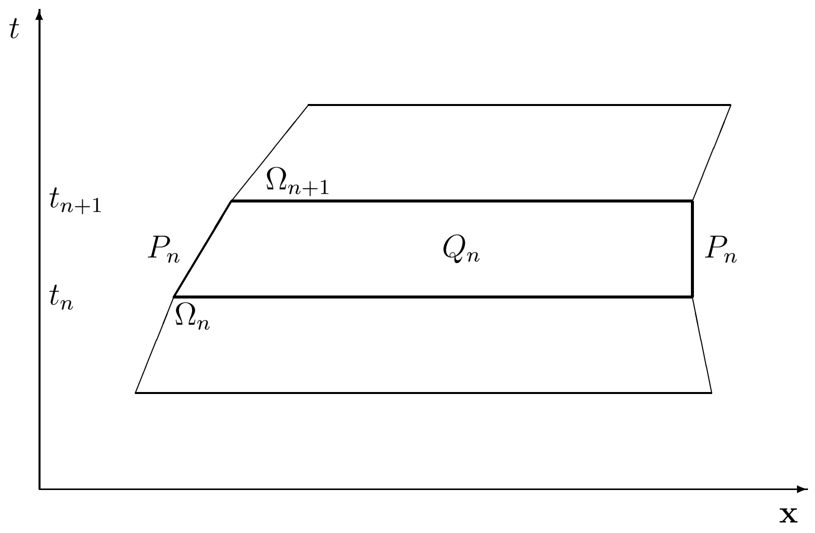

Figure 3.2. Space-time discretization: concept of a space-time slab.

surface flows are amenable to treatment with specialized methods, which may use specific assumptions about the flow to produce an approximate set of governing equations. As is the case with the Shallow Water Equations (SWE) [27], this modified problem is often significantly simpler to model than the original one. The simplified problem is in turn solved with a suitable numerical method. These approximations will not be considered here.

3.2 Variational Formulation

In order to construct the finite element function spaces for the space-time method, we partition the time interval $(0, T)$ into subintervals $I_n = (t_n, t_{n+1})$, where t_n and t_{n+1} belong to an ordered series of time levels $0 = t_0 < t_1 < \cdots < t_N = T$. Let $\Omega_n = \Omega_{t_n}$ and $\Gamma_n = \Gamma_{t_n}$. We will define the space-time slab Q_n as the domain enclosed by the surfaces Ω_n, Ω_{n+1}, and P_n, where P_n is the surface described by the boundary Γ_t as t traverses I_n. These concepts are sketched in Figure 3.2. As it is the case with Γ_t, surface P_n can be decomposed into $(P_n)_{\boldsymbol{g}}$ and $(P_n)_{\boldsymbol{h}}$ with respect to the type of boundary condition (Dirichlet or Neumann) being applied. For each space-time slab, we define the following finite element interpolation function spaces for the velocity and pressure:

$$(\mathcal{S}^h_{\mathbf{u}})_n = \left\{\mathbf{u}^h \mid \mathbf{u}^h \in \left[H^{1h}(Q_n)\right]^{n_{\mathrm{sd}}}, \mathbf{u}^h \doteq \boldsymbol{g}^h \quad \text{on } (P_n)_{\boldsymbol{g}}\right\}, \tag{3.1}$$

$$(\mathcal{V}^h_{\mathbf{u}})_n = \left\{\mathbf{u}^h \mid \mathbf{u}^h \in \left[H^{1h}(Q_n)\right]^{n_{\mathrm{sd}}}, \mathbf{u}^h \doteq \mathbf{0} \quad \text{on } (P_n)_{\boldsymbol{g}}\right\}, \tag{3.2}$$

$$(\mathcal{S}^h_{p})_n = (\mathcal{V}^h_p)_n = \left\{p^h \mid p^h \in H^{1h}(Q_n)\right\}. \tag{3.3}$$

Over the element domain, the interpolation is constructed by using first-order polynomials in space and, depending on our choice, zeroth- or first-order polynomials in time. Globally, the interpolation functions are continuous in space but discontinuous in time. However, for two-liquid flows, the solution and variational function spaces for pressure should include the functions which are discontinuous across the interface. It should be remarked that when required, the method can employ higher order interpolation functions in both space and time, to provide necessary accuracy.

The stabilized space-time formulation for deforming domains can be written as follows: given $(\mathbf{u}^h)^-_n$, find $\mathbf{u}^h \in (\mathcal{S}^h_{\mathbf{u}})_n$ and $p^h \in (\mathcal{S}^h_p)_n$ such that $\forall \mathbf{w}^h \in (\mathcal{V}^h_{\mathbf{u}})_n$, $\forall q^h \in (\mathcal{V}^h_p)_n$:

$$\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}\right) dQ + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) dQ \\
+ &\int_{Q_n} q^h \nabla \cdot \mathbf{u}^h dQ + \int_{\Omega_n} (\mathbf{w}^h)^+_n \cdot \rho \left((\mathbf{u}^h)^+_n - (\mathbf{u}^h)^-_n\right) d\Omega \\
+ &\sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \tau_{\mathrm{MOM}} \frac{1}{\rho} \left[\rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h\right) - \nabla \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h)\right] \\
&\cdot \left[\rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}\right) - \nabla \cdot \boldsymbol{\sigma}(p^h, \mathbf{u}^h)\right] dQ \\
+ &\sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \tau_{\mathrm{CONT}} \nabla \cdot \mathbf{w}^h \, \rho \nabla \cdot \mathbf{u}^h dQ = \int_{(P_n)_{\boldsymbol{h}}} \mathbf{w}^h \cdot \boldsymbol{h}^h dP.
\end{aligned} \tag{3.4}$$

The notational conventions used in (3.4) are shown below:

$$(\mathbf{u}^h)^\pm_n = \lim_{\varepsilon \to 0} \mathbf{u}(t_n \pm \varepsilon), \tag{3.5}$$

$$\int_{Q_n} \ldots dQ = \int_{t_n} \int_{\Omega^h_t} \ldots d\Omega dt, \tag{3.6}$$

$$\int_{P_n} \ldots dP = \int_{t_n} \int_{\Gamma^h_t} \ldots d\Gamma dt. \tag{3.7}$$

The solution to (3.4) is obtained sequentially for all space-time slabs $Q_1, Q_2, \ldots, Q_{N-1}$. The computations start with

$$(\mathbf{u}^h)_0^+ = \mathbf{u}_0. \tag{3.8}$$

In the variational formulation given by (3.4), the first three terms of the left hand side, together with the right hand side, constitute the standard Galerkin formulation of the problem. The fourth integral enforces, in a weak sense, the continuity of the velocity in time over the lower boundary Ω_n of the space-time slab Q_n. The fifth term in (3.4) is a least-squares form of the momentum equation added to the formulation. This term provides crucial stability characteristics and will be discussed in more detail, including the definition of τ_{MOM}, in the following section. At high Reynolds numbers, the stability is further enhanced by incorporating the sixth term into formulation (3.4), which includes a least-squares form of the continuity equation. The coefficient τ_{CONT} will be defined in the next section as well.

The deformation of the mesh is reflected in the deformation of space-time elements, and affect directly the computation of the transport terms. This effect will be illustrated for a typical deformed space-time element shown in Figure 3.3. Let $\boldsymbol{\xi}$ and

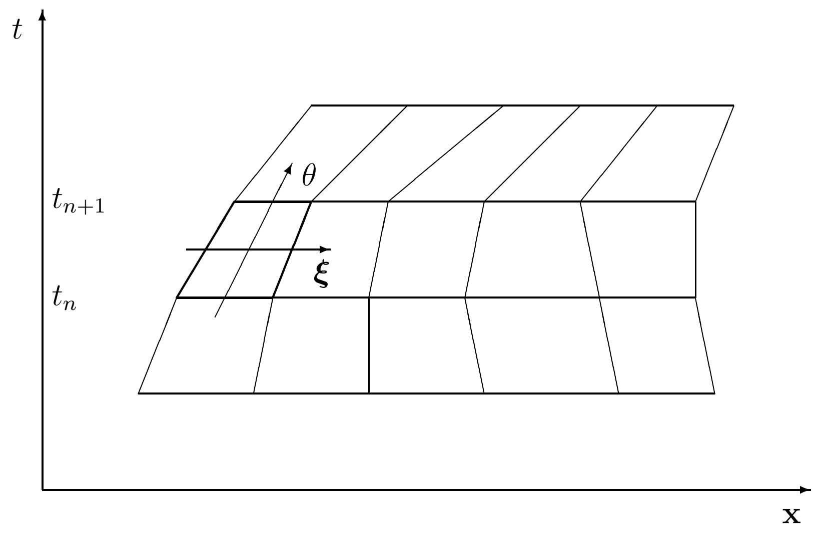

Figure 3.3. Space-time discretization: deformed element.

θ serve as a set of standard reference coordinates for that element. In the special, yet common, case when the $\theta = const$ levels coincide with the $t = const$ levels, the Jacobian matrix of the transformation between the reference and physical domains can be written as:

$$\frac{\partial(\mathbf{x}, t)}{\partial(\boldsymbol{\xi}, \theta)} = \begin{bmatrix} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} & \frac{\partial \mathbf{x}}{\partial \theta} \\ 0 & \frac{\partial t}{\partial \theta} \end{bmatrix}, \tag{3.9}$$

and its inverse as:

$$\frac{\partial(\boldsymbol{\xi}, \theta)}{\partial(\mathbf{x}, t)} = \begin{bmatrix} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}} & \frac{\partial \boldsymbol{\xi}}{\partial t} \\ 0 & \frac{\partial \theta}{\partial t} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}} & -\mathbf{v}^h \cdot \nabla \boldsymbol{\xi} \\ 0 & \frac{2}{\Delta t} \end{bmatrix}, \tag{3.10}$$

where $\Delta t = t_{n+1} - t_n$ and $\mathbf{v}^h$ is the local mesh velocity defined as:

$$\mathbf{v}^h = \left. \frac{\partial \mathbf{x}}{\partial t} \right|_{\theta = const}. \tag{3.11}$$

Now the transport term can be computed as:

$$\begin{aligned} \frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h &= \frac{\partial \mathbf{u}^h}{\partial \theta}\frac{\partial \theta}{\partial t} + \frac{\partial \mathbf{u}^h}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h \\ &= \frac{\partial \mathbf{u}^h}{\partial \theta}\frac{2}{\Delta t} - \frac{\partial \mathbf{u}^h}{\partial \boldsymbol{\xi}} \mathbf{v}^h \cdot \nabla \boldsymbol{\xi} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h \\ &= \frac{\partial \mathbf{u}^h}{\partial \theta}\frac{2}{\Delta t} + \left(\mathbf{u}^h - \mathbf{v}^h\right) \cdot \nabla \mathbf{u}^h. \end{aligned} \tag{3.12}$$

This is equivalent to modifying the advective velocity by subtracting the mesh velocity. If the element does not deform, as in a fully Eulerian description, the $\frac{\partial \boldsymbol{\xi}}{\partial t}$ term vanishes and the transport terms retain full advective velocity. On the other hand, in an approximately Lagrangian description, the node movement follows the fluid velocity field, and the effective advection velocity $\left(\mathbf{u}^h - \mathbf{v}^h\right)$ is close to zero.

3.3 Stabilization Details

The mixed Galerkin formulation, i.e., formulation (3.4) with $\tau_{\text{MOM}} = 0$ and $\tau_{\text{CONT}} = 0$, suffers from two stability problems. The first of the two is the difficulty encountered when one attempts to obtain numerical solution to advection-dominated problems with a pure Galerkin method. It is now well known that the Galerkin method applied to an advection-diffusion problem (or central difference approximation in the finite difference context) exhibits a level of diffusion lower than the physical problem. Defining a numerical diffusion as the diffusion change from the actual to the approximated system, we speak of a negative numerical diffusion of a Galerkin approximation. If not overcome by the sufficiently high level of actual diffusion in the problem, the Galerkin approximation may acquire negative total diffusion, producing spurious and growing oscillations, appearing in the vicinity of boundary layers and polluting the entire domain. The situation may be remedied either by dramatic mesh refinement, or by adding a certain amount of artificial diffusion to the formulation. The second option is generally accepted today, since methods exist to determine nearly optimal artificial diffusion, and moreover, to apply it properly in multi-dimensional advective systems. The streamline upwind/Petrov-Galerkin (SUPG) technique introduced by Brooks and Hughes [28] has been widely accepted in fluid dynamics community, and is a part of the stabilization terms of (3.4). See [28] also for a more detailed discussion of streamline upwind methods, culminating in SUPG.

The second stability problem is unique to mixed methods, and manifests itself for only certain combinations of the interpolations for the velocity and pressure fields. Simply stated, the danger stems from the fact that the pressure field enters the mixed Galerkin variational formulation (3.4 with $\tau_{\mathrm{MOM}} = 0$) only through the term $\int_{Q_n} \nabla \cdot \mathbf{w}^h p^h dQ$. For not sufficiently rich $(\mathcal{V}^h_{\mathbf{u}})_n$, spurious pressure fields, or modes, may "slip through" a net of velocity weighting functions and remain undetected by the variational formulation. The presence of spurious pressure modes is precluded by the inf-sup condition, due to Babuška and Brezzi, which may be written as follows:

$$\sup_{0 \neq \mathbf{u} \in (\mathcal{S}^h_{\mathbf{u}})_n} \frac{(\nabla \cdot \mathbf{u}, p)}{\| \mathbf{u} \|_1} \geq C \| p \|_0, \quad p \in (\mathcal{S}^h_p)_n. \tag{3.13}$$

Unfortunately, many desirable combinations do not satisfy (3.13), and efforts have been made to circumvent the inf-sup condition by modifying the Galerkin formulation itself. This has been accomplished for the Stokes problem by Brezzi and Pitkäranta [29]. A consistent approach for the Stokes problem was introduced in [30].

Here the weighted residual character of the Galerkin formulation was preserved, but the weighting functions themselves were modified. A generalization of that work for the finite Reynolds number flows led to pressure stabilized/Petrov-Galerkin (PSPG) formulation developed by Tezduyar et al. [31]. As seen in [31], the PSPG formulation has been successfully applied to different equal-order interpolations in the context of Navier-Stokes equations.

The Galerkin/Least-Squares technique used here combines the SUPG and PSPG stabilizations in one conceptually simple term. As seen in (3.4), the addition to the weighting function has the form of the variation of the momentum equation.

The τ_{CONT}-term has been proposed by Franca and Hughes [32], and provides stability to the velocity field at high Reynolds numbers. Such term can dramatically enhance the convergence of the nonlinear iterative algorithm, without compromising the consistency of the original method. The standard stability analysis for the advection-diffusion equation with SUPG or Galerkin/Least-Squares stabilization relies on the divergence-free property of the advection velocity field. In general, the velocity field obtained by solving (3.4) will become free of divergence only in the $h \rightarrow 0$ limit, where h is the mesh parameter. This potential for instability is eliminated with the addition of the consistent τ_{CONT}-term to the formulation.

3.3.1 Parameter Design

A parameter τ_{MOM} designed specifically for use with bilinear interpolations has been given in, e.g., [19], and is repeated here:

$$\tau_{\text{MOM}} = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{2|\mathbf{u}^h|_2}{h_e} \right)^2 + \left(\frac{4\nu}{h_e^2} \right)^2 \right]^{-1/2}, \tag{3.14}$$

or

$$\tau_{\text{MOM}} = \left[\left(\frac{2|\mathbf{u}^h|_2}{h_e} \right)^2 + \left(\frac{4\nu}{h_e^2} \right)^2 \right]^{-1/2}, \tag{3.15}$$

in the steady-state case. Here $|\mathbf{u}^h|_p = \left(\sum_{d=1}^{n_{\text{sd}}} |u_d^h|^p \right)^{1/p}$ is the pointwise velocity norm, h_e is the element length and $\nu = \mu/\rho$ is the kinematic viscosity. The second design (3.15) coincides in the advective limit with a definition used earlier for non-space-time formulations (see e.g. [31]):

$$\begin{aligned} Re_e &= \frac{|\mathbf{u}^h| h_e}{2\nu}, \\ \xi(Re_e) &= \begin{cases} Re_e/3, & Re_e < 3 \\ 1, & Re_e \geq 3 \end{cases}, \\ \tau_{\text{MOM}} &= \frac{h_e}{2|\mathbf{u}^h|_2}\xi(Re_e), \end{aligned} \tag{3.16}$$

where Re_e is an element level Reynolds number.

Another possible definition of τ_{MOM}, designed to be applicable to <u>any</u> order of interpolation, follows the definition given in [33], and may be summarized as follows:

$$\begin{aligned} Re_e &= \frac{m_e|\mathbf{u}^h|_2 h_e}{4\nu}, \\ \xi(Re_e) &= \begin{cases} Re_e, & Re_e < 1 \\ 1, & Re_e \geq 1 \end{cases}, \\ \tau_{\text{MOM}} &= \frac{h_e}{2|\mathbf{u}^h|_2}\xi(Re_e), \\ m_e &= \min\left\{\frac{1}{3}, 2C_I\right\}, \\ C_I \sum_{e=1}^{(n_{el})_n} h_e^2 \|\nabla\cdot\boldsymbol{\varepsilon}(\mathbf{w}^h)\|_{0,e}^2 &\leq \|\boldsymbol{\varepsilon}(\mathbf{w}^h)\|_0^2 \qquad \forall \mathbf{w}^h \in (\mathcal{V}_{\mathbf{u}}^h)_n. \end{aligned} \tag{3.17}$$

Constant m_e accounts for different orders of interpolation and depends on the inverse estimate constant C_I. For bilinear velocity interpolations, $C_I = 0$ and $m_e = 1/3$. This design will be used with the velocity-pressure-stress formulation examples in Chapter 6.

The definition of τ_{CONT} is the same as the one introduced in [34]:

$$\tau_{\text{CONT}} = \vartheta|\mathbf{u}^h|_2 h_e \xi(Re_e), \tag{3.18}$$

with Re_e and $\xi(Re_e)$ defined as in (3.16) if τ_{MOM} definitions (3.14) or (3.16) are used. Alternatively we define Re_e and $\xi(Re_e)$ as it is done in (3.17) if that is the definition used for τ_{MOM}. Here ϑ is a positive parameter, usually taken as unity. The definition (3.18) is a smoothed version of the parameter initially used in [33].

3.3.2 Low Order Elements

Special remarks will apply to the formulation (3.4) when it is used with first-order interpolation functions for the velocity field. Although the formulation appears to be consistent, the least squares stabilization terms involve an unmodified form of the momentum equation, and hence second derivatives of the velocity field. These derivatives vanish identically for linear (triangle) velocity interpolations, and are incomplete for bilinear (quadrilateral) velocity elements. Thus, while the full viscous momentum equation is represented in the Galerkin part, only an inviscid version of it is present in the least-squares contribution. The imbalance created by the absence of the viscous terms will be normally absorbed by the pressure and convective terms. This effect is hardly noticeable for the τ_{MOM} definitions in this section. But in the space-time formulation, the presence of the time evolution terms in the least-squares contribution may lead to a non-negligible and surprising phenomenon. When a steady flow of a fluid is computed with a time-dependent procedure, the convergence is achieved when $(\mathbf{u}^h)^-_{n+1} = (\mathbf{u}^h)^-_n$, yet $(\mathbf{u}^h)^-_n$ need not be equal to $(\mathbf{u}^h)^+_n$. When this criterion is used, there exists the possibility of a non-zero $\partial \mathbf{u}^h/\partial t$ and a corresponding non-zero jump-term. Indeed, the already mentioned inconsistency seen in the least-squares form of the momentum equation, will feed such saw-tooth pattern in the time behavior of the solution. Moreover, the smaller the time step, the larger part of the inconsistency will be absorbed by the $\partial \mathbf{u}^h/\partial t$ within a space-time slab, instead of being absorbed by the remaining spatial (pressure and convective) terms of the momentum equation. More importantly, by varying the time step size, the weighting expression $\partial \mathbf{w}^h/\partial t$ is also changed, leading to relative shifts in weight between the Galerkin and Least-Squares forms of the momentum equation. This accounts for the visible dependence of the steady-state solution on the time step, when the bilinear velocity interpolation is used. Since the stress-velocity-pressure formulation discussed in Chapter 4 is free of inconsistency even for lowest order interpolations, it is expected to be also free of such unwanted effects in its space-time implementation. Of course that formulation also involves extra cost compared to the velocity-pressure formulation.

Another low order element anomaly appears in the context of temporal discretization. In the applications of space-time methods to the fixed domain problems, a satisfactory solutions may be obtained with the use of constant-in-time interpolation functions. When such interpolation is applied to the deforming domain problems, it falls into the subparametric category, which may invalidate some of the assumptions of convergence analysis, such as completeness of the shape function sets.

3.4 Moving Boundary Treatment

Similarly to other Lagrangian-Eulerian formulations, the current formulation gives us nearly complete freedom of the mesh movement within the domain, and to some extent also at the boundary. In the particular case of free surfaces and moving no-flux interfaces, the only requirement for the velocity $\mathbf{v}$ of the nodes on that boundary is $\mathbf{v}\cdot\mathbf{n} = \mathbf{u}\cdot\mathbf{n}$, i.e., the normal components of the nodal and fluid velocities must match. The tangential component of $\mathbf{v}$ remains arbitrary, with some common choices depicted in Figure 3.4. Boundary nodes can be made to move in a prescribed direction (e.g. horizontal or vertical), in a direction normal to the free surface, or in the direction of local fluid velocity vector. In the last case we obtain a locally Lagrangian description at the boundary. In all moving boundary problems described here, the node motion in the normal direction only is selected. Movement of the mesh inside the domain is the subject of the next subsection.

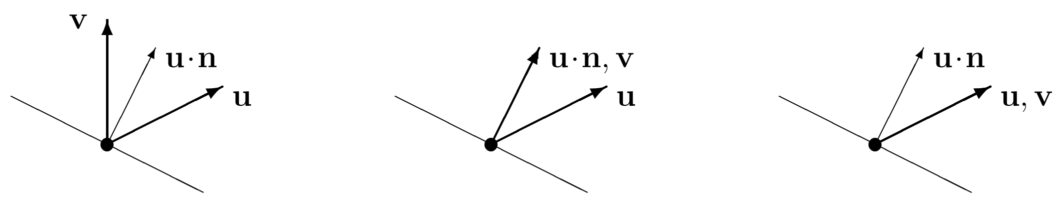
$\mathbf{v}$
$\mathbf{u}\cdot\mathbf{n}$
$\mathbf{u}$
$\mathbf{u}\cdot\mathbf{n}, \mathbf{v}$
$\mathbf{u}$
$\mathbf{u}\cdot\mathbf{n}$
$\mathbf{u}, \mathbf{v}$

Figure 3.4. Mesh moving options at the free surface: prescribed direction (left), normal direction (center) and local velocity direction (right).

3.4.1 Mesh Moving Options

In problems involving mild or cyclic deformations only, the initial mesh can be designed to provide the necessary flexibility, and remain topologically unchanged throughout the simulation. In problems involving large deformations, on the other hand, it may be necessary to introduce new meshes during the course of the computations, and project the solution from the old set of nodes to the new one. There are situations in which the deformation of the mesh is known in advance, either precisely, or approximately. Such is the case when computing flows around bodies moving in a prescribed manner. Thus the initial mesh can be made to absorb certain types of deformations without significant element distortion or the need for remeshing, as shown schematically in Figure 3.5. This approach is successfully used in conjunction with

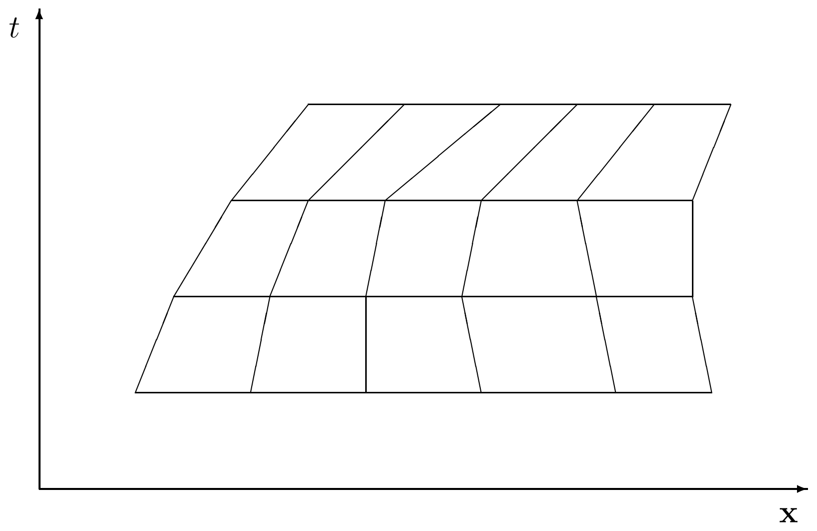

Figure 3.5. Mesh moving options: movement with no remeshing.

the present formulation to compute flows past oscillating cylinders and pitching airfoils (see [23,24]). In general, such techniques will be successful when the deformation of the domain is either cyclic or tends to a steady-state.

In many cases, however, the distortion of the domain is irregular or unpredictable, and therefore the design of an explicit mesh moving method is difficult. More flexible techniques are being developed, in which the domain is treated as an elastic solid under given boundary displacements [22]. The movement of the nodes is determined by solving the elasticity equations to obtain the displacement field in the interior of the domain. The shape of the critical elements can be improved by adjusting the (fictitious) material properties for these elements. Only the initial mesh needs to be explicitly generated, possibly by an automatic mesh generator.

Still, in some other cases, dramatic changes in the shape and volume of the domain may cause both of the aforementioned techniques to break down. Our approach here is to use an automatic mesh generator, the INRIA-developed Emc^2 [35], to discretize anew the domain deformed during the previous time step, as shown in Figure 3.6. The solution has to be projected to the new mesh via an interpolation technique. The

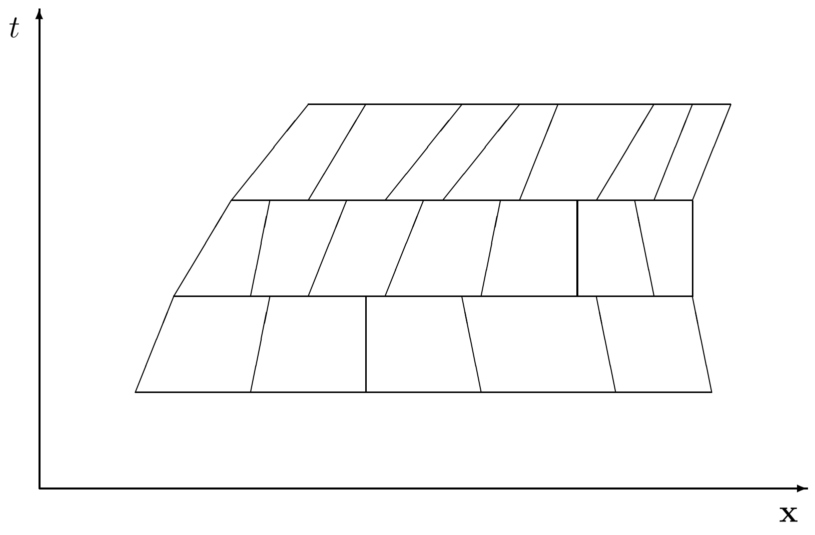

Figure 3.6. Mesh moving options: movement with discrete remeshing.

jump term integral in (3.4) can also serve as a projection mechanism. The ability of this scheme to handle arbitrary domains is offset by the diffusive effect of the projection, which grows with increasing time resolution, as the number of projections is increasing with decreasing time step size.

The ideal approach should integrate the elastic treatment of the interior domain, with infrequent remeshing when unacceptable mesh distortions are reached. It is also desirable to isolate, if possible, a section of the domain in which remeshing should be avoided, and preserve the mesh in this region even as the mesh in the rest of the computational domain undergoes restructuring and data there is projected. It is clear that the linear projection will not affect a flow field with uniform velocity or uniform shear rate. On the other hand the adverse effects of projection are most pronounced in regions where the shear rate varies rapidly e.g. in boundary layers or internal shear layers. Sometimes, the location of such regions is predictable. For example, computation of fluid flow around a solid object might involve a structured, stiff mesh fragment around the object which is never subjected to remeshing and the associated projection process, thus preserving the flow field within its boundary layer. Outside of this small area, even frequent remeshing may be allowed, as it will not degrade the quality of a slowly varying flow field.

Yet another possibility is available to us with the use of a completely unstructured mesh within the space-time slab. In the approaches discussed so far, the space-time mesh was formed by a straightforward extension of the spatial mesh in the time dimension. Such strategy yields three-dimensional six-node or eight-node bricks for two-dimensional problems discretized with triangular or quadrilateral elements, and four-dimensional bricks for three-dimensional problems. The important point is that the mesh generation burden is restricted to the spatial domain only. With n_{sd}-simplex mesh in the spatial domain, it is also possible to connect two dissimilar meshes at the two time levels of a space-time slab with an unstructured $(n_{\mathrm{sd}}+1)$-simplex mesh. The remeshing in this case is achieved in a continuous manner, as shown in Figure 3.7. Unfortunately with the present state of affairs in the area of mesh generation, this strategy is not yet viable for three-dimensional problems. The mesh motion with no remeshing (Figure 3.5) is utilized in the sloshing problem in Section 6.1. To illustrate the discrete remeshing strategy (Figure 3.6) we discuss now a simple problem involving large deformation of the domain and its boundary.

Fountain Flow

A flow from a fountain has been one of the numerous problems solved in [2] with the MAC method, and recently used by Hansbo [25]. In this experiment, the fluid enters a short vertical pipe as shown in Figure 3.8. The dimensions of the pipe are $W = 1.0$

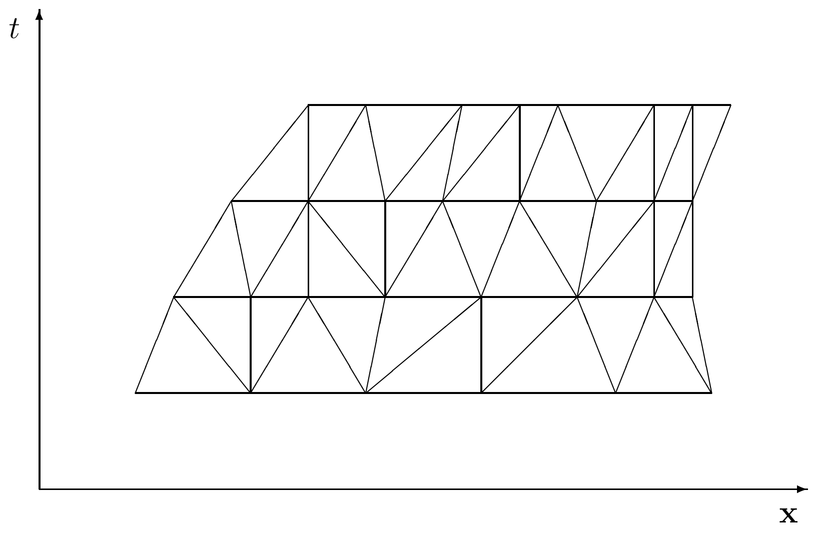

Figure 3.7. Mesh moving options: movement with continuous remeshing.

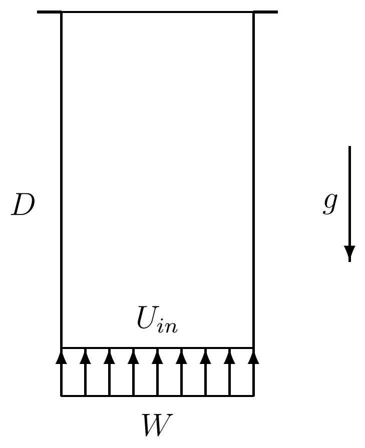

Figure 3.8. Fountain flow: domain description.

and $D = 2.0$. The gravitational acceleration is taken as $g = 1.0$, density as $\rho = 1.0$ and the fluid is assumed to be inviscid $\nu = 0$. The pipe walls admit slip, while the top boundary is traction-free and is moving with the fluid in the direction normal to the surface. The time step is chosen as $\Delta t = 0.05$. The fluid enters the domain at the bottom of the pipe with vertical velocity $U_{in} = 1.0$. Subsequently, it overflows the pipe walls and falls down under the influence of gravity forming symmetric tails extending downwards. After the initial development phase, the pressure inside the pipe reaches a new hydrostatic steady state, and tends to zero in the fountain tails. Also the shape of the domain in the vicinity of the pipe outlet achieves a time independent form. The evolution of the domain and its mesh is shown in Figures 3.9 and 3.10. Corresponding pressure fields are shown in Figures 3.11 and 3.12. The number of elements grows from the initial 614 to 2,126 at $t = 6.0$.

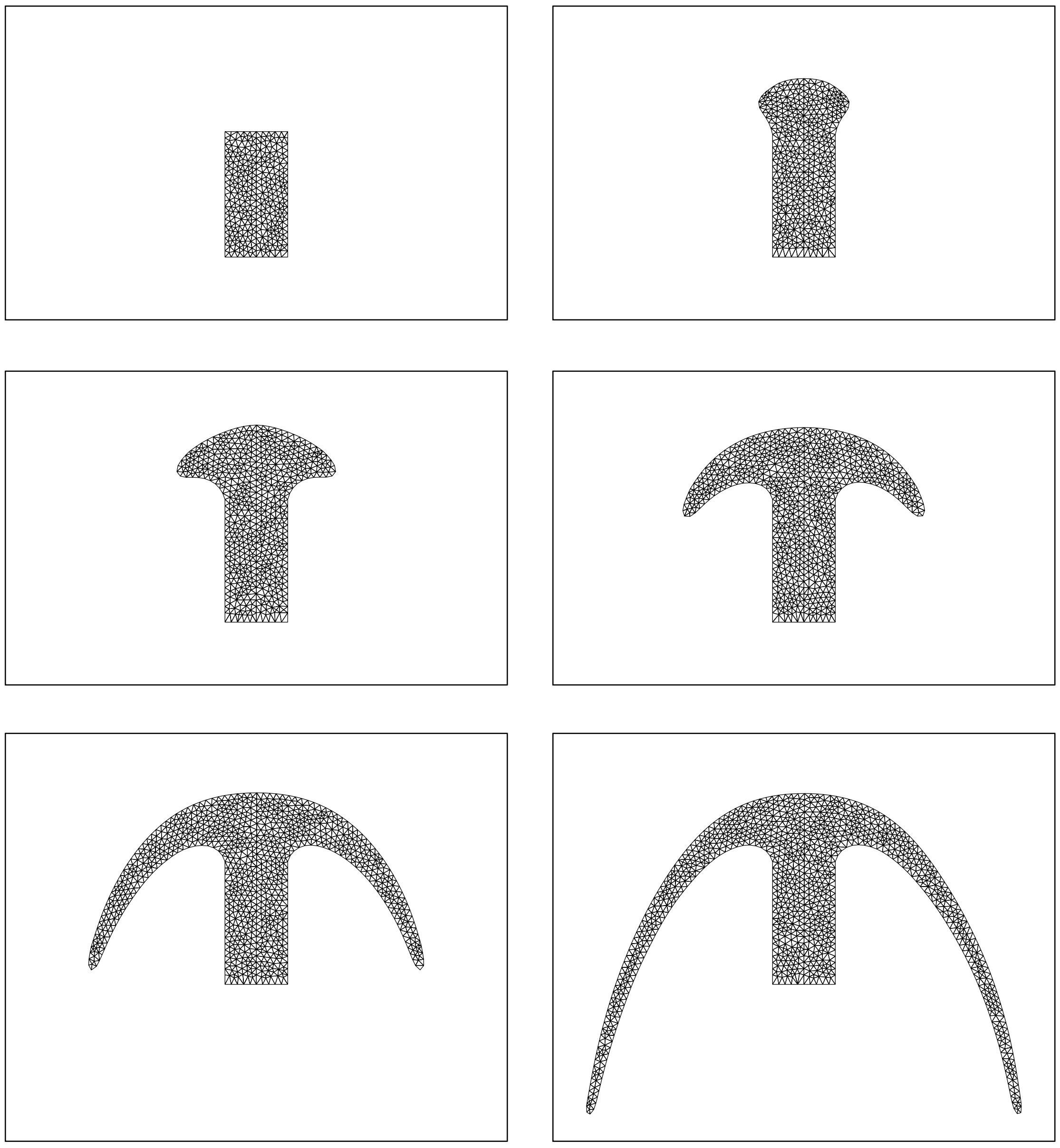

Figure 3.9. Fountain flow: finite element mesh at $t = 0.0, 1.0$ (top row), $t = 2.0, 3.0$ (middle row) and $t = 4.0, 5.0$ (bottom row).

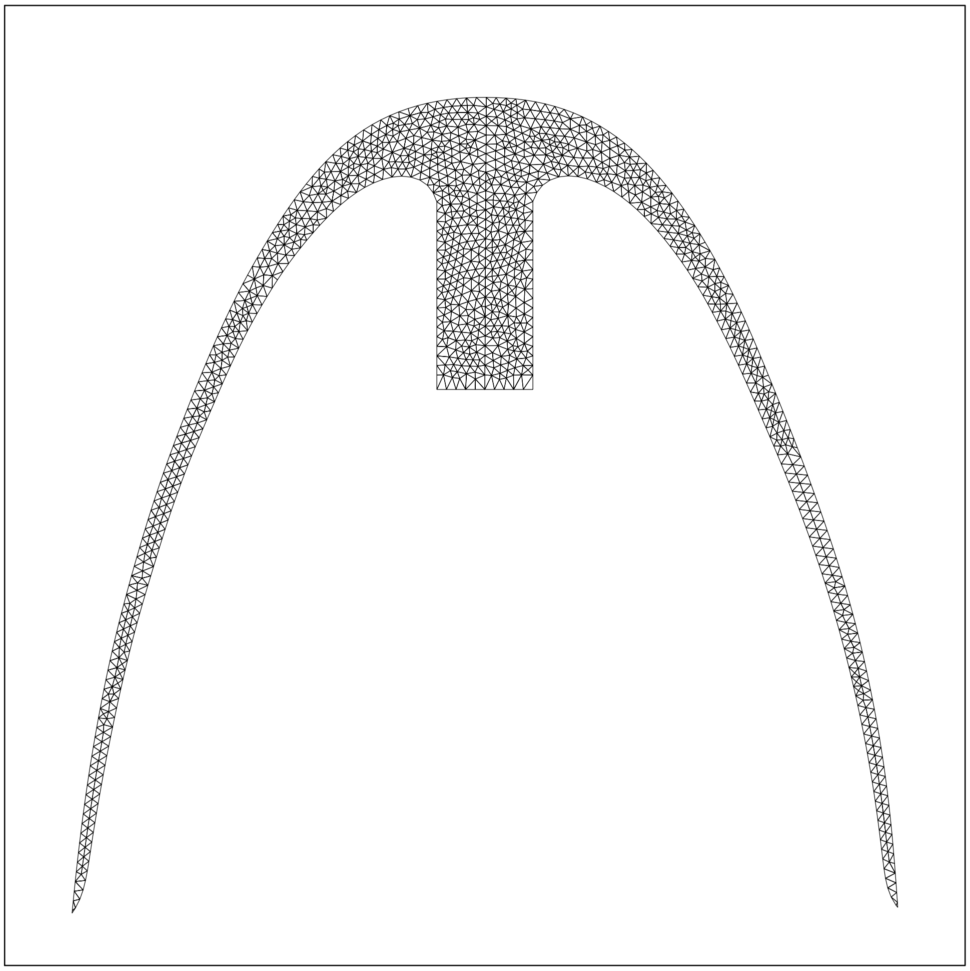

Figure 3.10. Fountain flow: finite element mesh at $t = 6.0$.

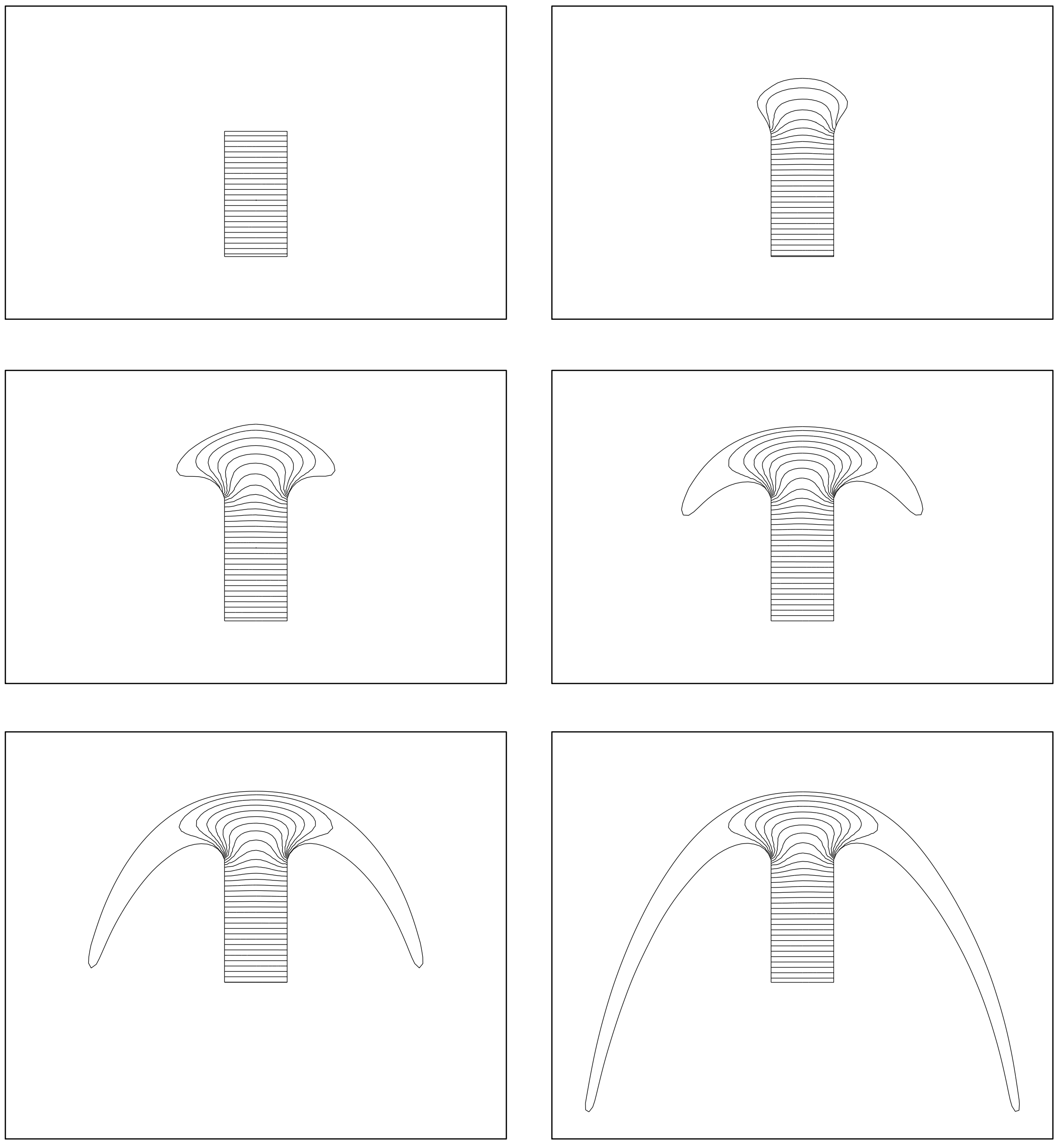

Figure 3.11. Fountain flow: pressure field at $t = 0.0, 1.0$ (top row), $t = 2.0, 3.0$ (middle row) and $t = 4.0, 5.0$ (bottom row).

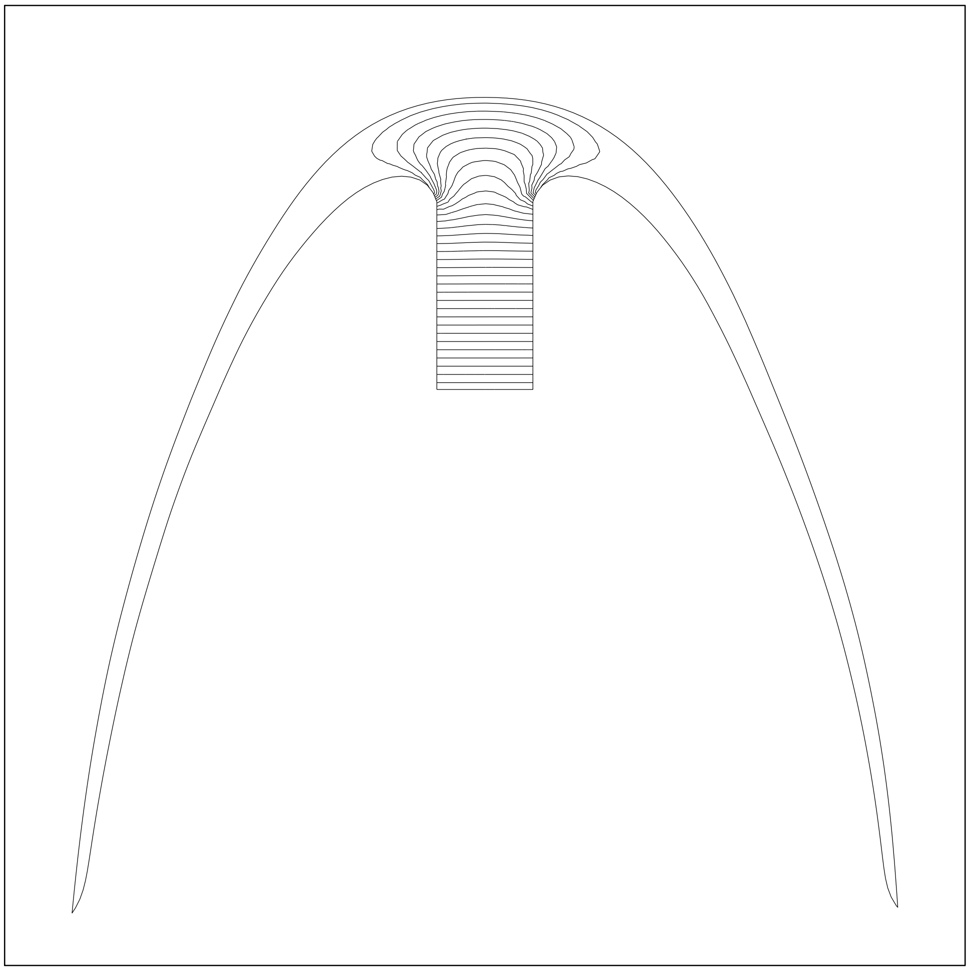

Figure 3.12. Fountain flow: pressure field at $t = 6.0$.

3.4.2 Surface Tension

The surface tension effects can play an important role in free-surface flows. Although the particular examples discussed in this thesis do not involve surface tension as a significant factor, other applications of the current method [22] include surface tension effects.

The surface tension mechanism is based on a change in the normal stress component at a free surface (or at an interface between two fluids). In two dimensions, this change is related to the radius of curvature as follows:

$$\mathbf{n} \cdot \boldsymbol{\sigma} \cdot \mathbf{n} = \frac{\gamma}{R} \qquad \text{on } (\Gamma_t)_{free}. \tag{3.19}$$

Here γ is the surface tension coefficient and R is the radius of curvature, defined to be positive when $\mathbf{n}$ points towards the center of curvature. Condition (3.19) is particularly simple to impose as it falls into the category of natural boundary conditions for the problem (2.1). In the formulation given by (3.4) we add a term

$$\int_{(P_n)_{free}} \mathbf{w}^h \cdot \mathbf{n} \frac{\gamma}{R}\, dP, \tag{3.20}$$

where $(P_n)_{free}$ is the part of the space-time slab surface P_n, traced by the moving boundary $(\Gamma_t)_{free}$.

3.5 Matrix Form

In order to discuss the implementational aspects, we need to describe in more detail the process of forming and solving the linear equation system from the abstract variational form (3.4).

To simplify the notation, we introduce a composite trial solution

$$\bar{\mathbf{u}}^h = \left\{ \bar{u}_i^h \right\}_{i=1,n_{\mathrm{dof}}} = \bar{u}_i^h \bar{\mathbf{e}}_i, \quad \bar{u}_i^h = \left\{ \begin{array}{ll} u_i^h & 1 \le i \le n_{\mathrm{sd}} \\ p^h & i = n_{\mathrm{sd}} + 1 \end{array} \right. , \tag{3.21}$$

where $n_{\mathrm{dof}} = n_{\mathrm{sd}} + 1$ is the number of degrees of freedom per node. In the presence of Dirichlet boundary conditions, the composite solution is decomposed into its known and unknown parts

$$\bar{\mathbf{u}}^h = \bar{\mathbf{v}}^h + \bar{\boldsymbol{g}}^h, \quad \bar{\mathbf{v}}^h = \bar{v}_i^h \bar{\mathbf{e}}_i, \quad \bar{\boldsymbol{g}}^h = \bar{g}_i^h \bar{\mathbf{e}}_i, \tag{3.22}$$

with

$$\bar{v}_i^h = \begin{cases} v_i^h & 1 \le i \le n_{\text{sd}} \\ p^h & i = n_{\text{sd}} + 1 \end{cases}, \quad \bar{g}_i^h = \begin{cases} g_i^h & 1 \le i \le n_{\text{sd}} \\ 0 & i = n_{\text{sd}} + 1 \end{cases}. \tag{3.23}$$

Then the composite solution is represented in terms of basis (shape) functions:

$$\bar{v}_i^h = \sum_{A \in \eta - \eta_{g_i}} N_A d_{iA}, \quad \bar{g}_i^h = \sum_{A \in \eta_{g_i}} N_A g_{iA}, \tag{3.24}$$

where η and η_{g_i} represent the set of all nodes, and the set of Dirichlet nodes for the degree of freedom i, respectively. Similar representation applies to the composite weighting function:

$$\bar{\mathbf{w}}^h = \bar{w}_i^h \bar{\mathbf{e}}_i, \quad \bar{w}_i^h = \sum_{A \in \eta - \eta_{g_i}} N_A c_{iA}. \tag{3.25}$$

The variational formulation (3.4) may be written in abstract form

$$a(\bar{\mathbf{w}}^h, \bar{\mathbf{v}}^h + \bar{\boldsymbol{g}}^h) = (\bar{\mathbf{w}}^h, \bar{\boldsymbol{h}}^h)_{P_n}, \tag{3.26}$$

where $\bar{\boldsymbol{h}}^h$ is $\boldsymbol{h}^h$ extended to the composite solution space in a manner analogous to $(3.23)_2$. Because of nonlinearity of the functional $a(\cdot\,,\cdot)$ in some of the components of the second vector argument, we apply an iterative Newton-Rhapson technique to obtain corrections to the current value $\bar{\mathbf{v}}^{h*}$:

$$\bar{\mathbf{v}}^h = \bar{\mathbf{v}}^{h*} + \Delta\bar{\mathbf{v}}^h, \quad a(\bar{\mathbf{w}}^h, \bar{\mathbf{v}}^h + \bar{\boldsymbol{g}}^h) = a(\bar{\mathbf{w}}^h, \bar{\mathbf{v}}^{h*} + \bar{\boldsymbol{g}}^h) + a_I(\bar{\mathbf{w}}^h, \Delta\bar{\mathbf{v}}^h), \tag{3.27}$$

where $a_I(\cdot\,,\cdot)$ is bilinear. Note that for a linear problem (e.g., Stokes equation), $a_I(\cdot\,,\cdot) = a(\cdot\,,\cdot)$. From (3.27) follows a decomposition of the nodal unknown vector introduced in (3.24):

$$d_{jB} = d_{jB}^* + \Delta d_{jB}, \tag{3.28}$$

and at each nonlinear iteration step we have to solve the following equation:

$$\sum_{j=1}^{n_{\text{dof}}} \left(\sum_{B \in \eta - \eta_{g_i}} a_I(N_A \mathbf{e}_i, N_B \mathbf{e}_j) \Delta d_{jB} \right) = (N_A \mathbf{e}_i, \bar{\boldsymbol{h}}^h)_{P_n} \quad - \quad a(N_A \mathbf{e}_i, \bar{\mathbf{v}}^{h*} + \bar{\boldsymbol{g}}^h), \qquad (3.29)$$
$$A \in \eta - \eta_{g_i}, \; 1 \leq i \leq n_{\text{dof}}.$$

Equation (3.29) may be written in a matrix form as follows:

$$\begin{aligned} &\mathbf{K}\Delta\mathbf{d} = \mathbf{F}, \quad \mathbf{K} = [K_{PQ}], \quad \Delta\mathbf{d} = \{\Delta d_Q\}, \quad \mathbf{F} = \{F_P\}, \\ &K_{PQ} = a_I(N_A \mathbf{e}_i, N_B \mathbf{e}_j), \quad F_P = (N_A \mathbf{e}_i, \bar{\boldsymbol{h}}^h)_{P_n} - a(N_A \mathbf{e}_i, \bar{\mathbf{v}}^{h*} + \bar{\boldsymbol{g}}^h), \\ &P = \mathtt{ID}(i, A), \quad Q = \mathtt{ID}(j, B), \quad A, B \in \eta - \eta_{\mathbf{g}_i}, \quad 1 \leq i, j \leq n_{\text{dof}}, \end{aligned} \qquad (3.30)$$

where ID is the mapping assigning equation numbers to the nodal degrees of freedom. In the standard finite element implementation, the global matrix **K** and vector **F** are assembled, or can be thought of as being assembled, from the element-level contributions:

$$\begin{aligned} &\mathbf{k}^e = \left[k^e_{pq}\right], \quad \mathbf{f}^e = \left\{f^e_p\right\}, \quad 1 \leq p, q \leq n_{\text{ee}}, \\ &k^e_{pq} = a_I(N_a \mathbf{e}_i, N_b \mathbf{e}_j)^e, \quad f_p = (N_a \mathbf{e}_i, \bar{\boldsymbol{h}}^h)_{P^e_n} - a(N_a \mathbf{e}_i, \bar{\mathbf{v}}^{h*} + \bar{\boldsymbol{g}}^h)^e, \\ &p = n_{\text{dof}}(a-1) + i, \quad q = n_{\text{dof}}(b-1) + j, \quad 1 \leq a, b \leq n_{\text{en}}, \quad 1 \leq i, j \leq n_{\text{dof}}, \end{aligned} \qquad (3.31)$$

where n_{en} is the number of nodes in a space-time element, $n_{\text{ee}} = n_{\text{dof}} n_{\text{en}}$ is the number of element equations (or degrees of freedom), while N_a and N_b denote the restrictions of the shape functions to the local elemental space-time domains. Similarly the superscript e denotes restriction of the integral forms to the single element domain.

To demonstrate the preceding process in more detail, consider the first term in the weak form (3.4), namely

$$\int_{Q_n} \mathbf{w}^h \cdot \rho \frac{\partial \mathbf{u}^h}{\partial t} dQ. \qquad (3.32)$$

The corresponding contribution to the Galerkin form (3.26) is

$$a_t(\bar{\mathbf{w}}^h, \bar{\mathbf{v}}^h + \bar{\boldsymbol{g}}^h) = \int_{Q_n} \rho \bar{\mathbf{w}}^h \, \mathbf{I}^{\mathbf{uu}} \, \frac{\partial(\bar{\mathbf{v}}^h + \bar{\boldsymbol{g}}^h)}{\partial t} dQ, \qquad (3.33)$$

where

$$\mathbf{I}^{\mathbf{uu}} = \begin{bmatrix} \mathbf{I}_{n_{\text{sd}}\times n_{\text{sd}}} & \mathbf{0}_{n_{\text{sd}}\times 1} \\ \mathbf{0}_{1\times n_{\text{sd}}} & 0 \end{bmatrix} \tag{3.34}$$

is used to <u>filter</u> out the pressure degree of freedom. Consequently, the contributions to $a_I(N_A\mathbf{e}_i, N_B\mathbf{e}_j)$ and $a(N_A\mathbf{e}_i, \bar{\mathbf{v}}^{h*} + \bar{\boldsymbol{g}}^h)$ in (3.29) become

$$\begin{aligned} a_{I,t}(N_A\mathbf{e}_i, N_B\mathbf{e}_j) &= \int_{Q_n^e} \rho N_A \frac{\partial N_B}{\partial t} \delta_{ij}^{\mathbf{uu}} dQ^e\,, \quad A, B \in \eta - \eta_{\mathbf{g}_i}, \qquad (3.35) \\ a_t(N_A\mathbf{e}_i, \bar{\mathbf{v}}^{h*} + \bar{\boldsymbol{g}}^h) &= \int_{Q_n^e} \rho N_A \frac{\partial N_B}{\partial t} \delta_{ij}^{\mathbf{uu}} (d_{jB}^* + g_{jB}) dQ^e\,, \quad A \in \eta - \eta_{\mathbf{g}_i}, \quad B \in \eta, \end{aligned}$$

where $1 \leq i, j \leq n_{\text{dof}}$ and $\delta_{ij}^{\mathbf{uu}} = \mathbf{I}_{ij}^{\mathbf{uu}}$. Hence, $a_{I,t}(N_A\mathbf{e}_i, N_B\mathbf{e}_j) = 0$ when $i = n_{\text{dof}}$ or $j = n_{\text{dof}}$, consistent with (3.33). The contribution to $\mathbf{k}^e$ then takes the form

$$k_{pq}^{e,t} = a_{I,t}(N_a\mathbf{e}_i, N_b\mathbf{e}_j)^e\,. \tag{3.36}$$

That is,

$$\mathbf{k}^{e,t} = \begin{bmatrix} \hat{\mathbf{k}}_{11}^{e,t} & \hat{\mathbf{k}}_{12}^{e,t} & \dots & \hat{\mathbf{k}}_{1n_{\text{en}}}^{e,t} \\ \hat{\mathbf{k}}_{21}^{e,t} & \hat{\mathbf{k}}_{22}^{e,t} & \dots & \hat{\mathbf{k}}_{2n_{\text{en}}}^{e,t} \\ \vdots & \vdots & & \vdots \\ \hat{\mathbf{k}}_{n_{\text{en}}1}^{e,t} & \hat{\mathbf{k}}_{n_{\text{en}}2}^{e,t} & \dots & \hat{\mathbf{k}}_{n_{\text{en}}n_{\text{en}}}^{e,t} \end{bmatrix}, \tag{3.37}$$

where

$$\hat{\mathbf{k}}_{ab}^{e,t} = \begin{bmatrix} \int_{Q_n^e} \rho N_a \frac{\partial N_b}{\partial t} dQ^e & 0 & 0 & 0 \\ 0 & \int_{Q_n^e} \rho N_a \frac{\partial N_b}{\partial t} dQ^e & 0 & 0 \\ 0 & 0 & \int_{Q_n^e} \rho N_a \frac{\partial N_b}{\partial t} dQ^e & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad 1 \leq a, b \leq n_{\text{en}}. \tag{3.38}$$

Normally, the integration is carried out with a numerical integration rule (see e.g. Section 3.8 of [36]). The example just discussed will be referred to in the next section.

3.6 Parallel Implementation

Implementation notes contained in this section follow the parallel program design outlined in [21]. We will restrict the discussion to distributed memory Single Instruction/Multiple Data (SIMD) architectures, such as the Connection Machine range of supercomputers. It should be noted that the latest models in this hardware family are also capable of sustaining the Multiple Instruction/Multiple Data (MIMD) type of control flow.

The starting point in the design of a massively parallel implementation is the decision regarding distribution of the variables among the processors of the computer. Since most of massively parallel architectures are distributed memory machines, the placement of the data is of paramount importance and can radically affect the performance of the implementation. We will assume that the architecture which is used has a developed concept of virtual processors, so that each of the data entries which are subject to a parallel operation may be assumed to reside on its own processor. In reality, each physical processor takes over the duties of several virtual processors.

We will use two primary data storage modes termed `ELEM` and `EQN`. The `ELEM` mode is used for storage of element level data, with one element and its n_{ee} degrees of freedom associated with exactly one virtual processor. On the other hand the `EQN` mode will hold variables at the level of the global equation system (3.30). These two arrangements are shown schematically in Figure 3.13 for a typical subset of a finite element mesh. The nodal data, coordinates, element-level properties and element level matrices $\mathbf{k}^e$ are stored in the `ELEM` mode. The increment vector $\Delta\mathbf{d}$, residual $\mathbf{F}$ and certain intermediate variables are kept in the `EQN` form.

The mapping between the two datasets is denoted $\texttt{LM} : \texttt{ELEM} \mapsto \texttt{EQN}$. Communication operation $\texttt{ELEM} \leftarrow \texttt{EQN}$ is called a gather, while movement of the data in the opposite direction, $\texttt{ELEM} \rightarrow \texttt{EQN}$, is known as a scatter. The scatter is usually coupled with a combining operation at the destination, such as addition or overwriting. Both gather and scatter may be implemented efficiently on the Connection Machine computers, provided they are done repeatedly with a static communication

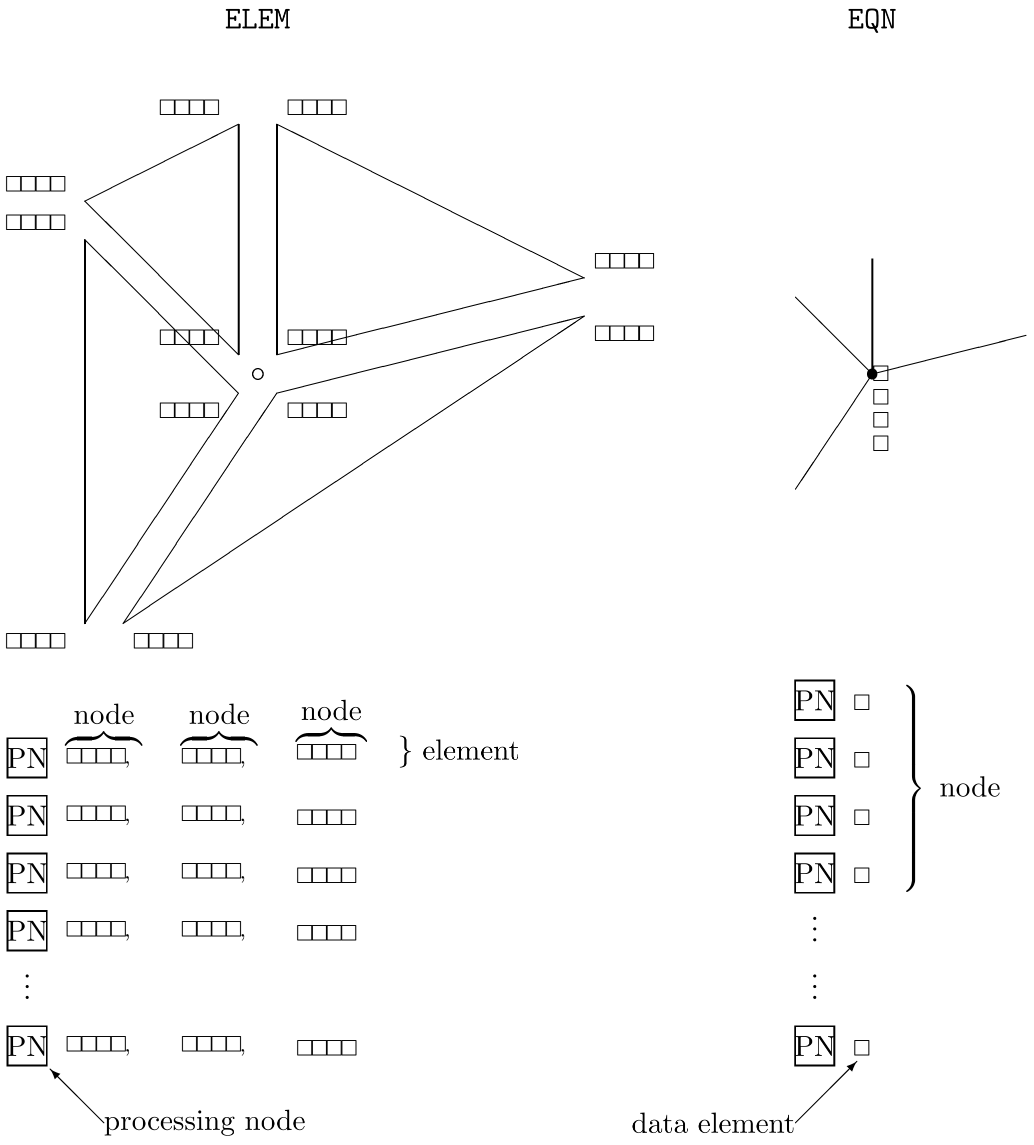

Figure 3.13. Parallel implementation: element-level (left) and equation-level (right) data storage modes.

pattern as defined by `LM`. In that case the communication trace may be saved the first time communication is performed, resulting in extremely fast subsequent gathers and scatters.

The process of solution of the linear system (3.30) which will be described in

Section 3.6, does not require the assembly of the global matrix $\mathbf{K}$ in any form. Instead, it operates on unassembled element level matrices $\mathbf{k}^e$. Therefore, apart from a simple assembly (scatter) of $\mathbf{F}$ from $\mathbf{f}^e$, the task of forming the equation system (3.30) takes place entirely at the element level. For example in the contribution to the element level stiffness matrix shown in (3.37) and (3.38), all quantities used to compute $\mathbf{k}^{e,t}$, i.e. ρ, N_a and $\partial N_b/\partial t$, as well as additional variables involved in numerical integration, are being computed and stored on the same virtual processor. Consequently, in the matrix formation phase, no inter-processor communication is involved and the parallelism of the operations may be fully exploited. Even more detailed description of the matrix formation phase, including a pseudocode fragment, is found in [21].